AS4 thoughts for PEPPOL

Author: Philip Helger, BRZ, Austria

# Introduction

PEPPOL is most likely switching from AS2 to AS4 and this document contains some general thoughts on it.

PEPPOL conceptually builds on the so called “4-corner model” and handles the business document exchange between corner 2 (C2) and corner 3 (C3). Corner 1 (original sender; C1) and Corner 4 (final receiver; C4) are out of scope of PEPPOL document exchange.

# AS2

The AS2 solution is using the following features:

* Basically an S/MIME message
* Requires https transport (transport level encryption) – using non-PEPPOL certificates (!)
* Business documents (UBL/XML) are wrapped in SBDH (Standard Business Document Header)
* AS2 transmissions are signed with the PEPPOL AP certificates

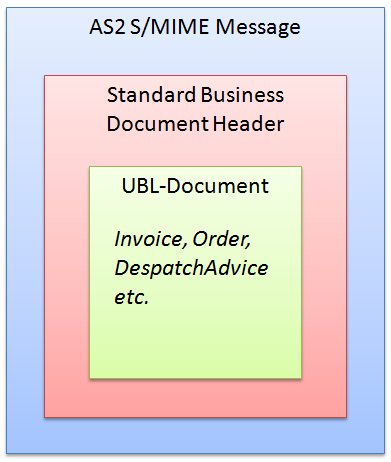


Figure 1: AS2 S/MIME document packaging

# AS4

AS4 messages either can be presented as SOAP messages or as MIME messages with a mandatory SOAP part (SOAP with attachments/MTOM). Both SOAP 1.1 and 1.2 are supported – SOAP 1.2 is the mandatory default.

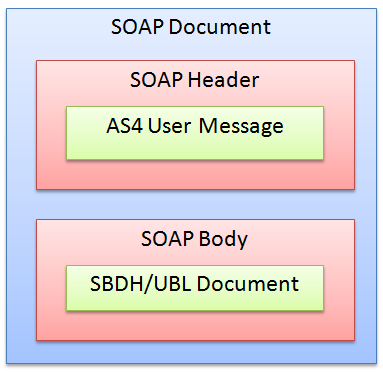


Figure 2: AS4 SOAP only packaging

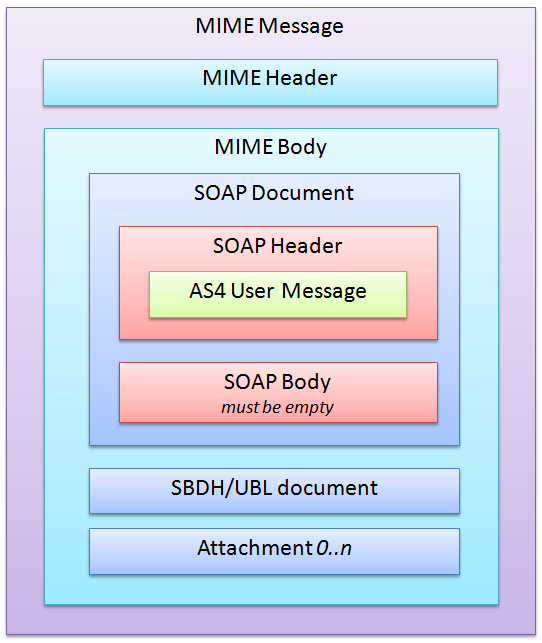


Figure 3: AS4 MIME packaging

If only the payload (the SBDH/UBL document) and no attachments are transmitted it can be packaged either as a SOAP document or as a MIME message. If attachments are present the MIME packaging must be used.

Message exchange patterns (MEP bindings) can be one of the following: push, pull, sync, push-push, push-pull, pull-push. “push” is the only “one way” pattern – all other patterns are “two way”.

# PEPPOL requirements

This chapter contains some thoughts on which AS4 design decisions might apply to PEPPOL in the future. This is work in progress and is not yet a final state!

## Document exchange choreography

The basic choreographies don’t consider MLR and/or BLR. See chapter 4.1.1 for details concerning MLR/BLR.

For a simple document exchange (as e.g. performed when using BIS 4A – Invoice only) the simple AS4 “one way” “push” MEP binding should be sufficient.

For more complex scenarios (like tenders or the Norwegian payment process) a “two way” message exchange may be needed but that needs to be defined (I simply don’t know). Potential divergent elements are:

* Business level signature needed?
* Different size limits – for an invoice 100MB should be enough for a catalogue this is just the beginning...
* Non-repudiation of receipt needed?
* End-to-end (C1 to C4) security needed?

TODO: PEPPOL PreACC and Norway

### Thoughts on “pull”

Using the AS4 “pull” MEP binding might be an option for all PEPPOL service providers still using the old (and deprecated) PEPPOL LIME (Lightweight Message Exchange) protocol.

I think it would be beneficial to specify a “pull” based protocol and let it be implemented on a voluntary basis.

### Correlation with MLR and BLR

MLR = Message Level Response (validation results)

BLR = Business Level Response (business level validation results)

We would need something like “push-push-push”:

* C2 🡪 C3 Document
* C3 🡪 C2 MLR (optional)
* C3 🡪 C2 BLR (optional)

Alternatively a “sync-push” MEP binding could be invented so that the first “sync” is the main document exchange as well as the (synchronous) MLR and the second “push” would be the optional BLR. The drawback is, that this would require a) a new MEP binding and b) a synchronous MLR.

## PMode considerations

A PMode (processing mode) defines the rules for document exchange between two parties. Usually PModes are configured statically between partners but in PEPPOL this needs to happen dynamically, as PEPPOL document exchange is per se dynamically and the big advantage is, that document exchange can happen without prior knowledge about all the potentially document exchange partners. That implies that a PEPPOL compliant AS4 solution must be able to dynamically create PModes on the fly while operating. More details on how SMP and PMode are correlated can be found in chapter 4.3.

It is recommended that each PMode that is dynamically created receives an ID that is a combination of the CN of the sender and the CN of the receiver separated by a dash (‘-‘). The “CN” is the “subject common name” part of the PEPPOL AP certificate (for example “APP\_1000000101”). So if “APP\_1000000101” is sending to “APP\_1000000202” the created PMode ID should be “APP\_1000000101-APP\_1000000202”.

The determination of the PMode on sender and/or receiver side is based on one of the following possibilities:

* Using special PMode Identifiers (CollaborationInfo/AgreementRef/@pmode)
  + This may work in scenarios where C2 and C3 previously exchanged messages but for a new combination of C2 and C3 this will most likely not work, except C2 and C3 apply the same logic for PMode identifiers as stated above
* Using a special combination of “CollaborationInfo/Service” and “CollaborationInfo/Action”
  + If these fields are filled with document type identifier and process identifier this may be used to determine the “PMode template” to be used. But depending on what exchange pattern is used (see above) there might be the necessity for more than one “PMode template”.
* All other possibilities (as. E.g. using data from the SBDH) is not AS4 compliant and might therefore lead to problems.

## SMP considerations

The SMP (Service Metadata Publisher) is the PEPPOL way to dynamically discover capabilities of partners. Currently an SMP contains only the information about receivers but this may change in the future. Some elements of an SMP query response (a so called “Endpoint”, based on the quadruple of participant ID, document type ID, process ID and transport profile) contains information that must be dynamically fed into a PMode:

* Endpoint URL (e.g. <https://test.e-rechnung.gv.at/as4>)
* The PEPPOL AP certificate (Base64 DER/PEM encoded)
  + The CN of this certificate is part of the PMode ID (see above)

Additionally the following elements of an SMP response must be taken into account

* Service activation and expiration date

The transport profile to be used is

bdxr-transport-ebms3-as4-v1p0

SMP information may change (endpoint URLs and or certificates) so AS4 software must be able to update existing PModes (if they are persistent).

## Payload considerations

In the current AS2 solution, the payload is an SBDH combined with the payload (UBL) document. The PEPPOL SBDH consists of the following information elements:

* PEPPOL Sender Participant ID (scheme and value)
* PEPPOL Receiver Participant ID (scheme and value)
* Exchanged document type ID (just the value – already bogus)
* Exchanged process ID (just the value – already bogus)
* Contained business document XML root element namespace URL and local element name (redundant as the payload is present anyway)
* An instance identifier (usually a UUID)
* Creation date and time

It may be an option to include these properties into the “user message” as “MessageProperties/Property” elements which would mean the SBDH could be spared. The issue here is to check, whether these properties are available in all exchanges or whether a complex scenario (like a tender) consists partially of “signal messages” only as only “pull message” “pull attributes” provide a comparable properties storage. “Error messages” and “receipts” only offer the possibility to define custom extensions (xs:any).

## Certificate usage

In the default PEPPOL document exchange three different certificates are used

* The https certificate for the secure document transport
  + As this certificate is not directly part of the PEPPOL transport infrastructure it can be ignored in this document. The same rules apply for AS2 and AS4 document exchange.
  + This certificate MUST NOT be self-signed.
  + It is recommended that this https certificate is issued by an internationally trusted issuer for maximum interoperability
* The PEPPOL SMP certificate for signing SMP responses
  + Must only be validated during a document exchange.
  + The signing happens in the SMP server and is out of scope of this document
* The PEPPOL AP certificate for signing the exchanged business document
  + The sending AP (C2) signs the document with his AP certificate
  + The receiving AP (C3) validates the signature of C2
  + If this is the first document exchange between C2 and C3 the AP certificate is not known to the receiver and therefore C2 (the sender) must always include his public certificate in the transmitted data so that C3 can validate the data and check that the signee was using the correct PEPPOL keystore.

This is true for both AS2 and AS4 document exchange.

TODO what if end-to-end security is needed

# Additional thoughts

This chapter contains random thoughts that may/should be added into this document

* Some AS4 solutions use REM evidences – according to Sander and my interpretation this could be avoided by making minor changes to the OASIS AS4 specifications
* The thoughts in the chapter should be aligned with the “PEPPOL delivery extension” work group – especially on the usage of ASiC containers.
* Size limitations as in eSENS must be discussed (2GB/4GB).
* Conformance Testing (similar to the one offered by CEF but with PEPPOL focus should be offered)